UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/677,129	09/30/2003	Pierre-Jean Reissman	13768.783.108	3890

47973 7590 11/02/2006

WORKMAN NYDEGGER/MICROSOFT
1000 EAGLE GATE TOWER
60 EAST SOUTH TEMPLE
SALT LAKE CITY, UT 84111

EXAMINER	
PHAM, THAI V	

ART UNIT	PAPER NUMBER
2191	

DATE MAILED: 11/02/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	10/677,129	REISSMAN ET AL.	
	Examiner	Art Unit	
	Thai Van Pham	2191	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *9/30/2003*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-35* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-35* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *30 September 2003* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

This is the initial office action based on the application filed on October 27, 2006.

Claims 1 – 35 are currently pending and have been considered below.

Specification

1. The disclosure is objected to because of the following informalities: typographical error(s).

-- On line 1 of page 21 in the Specification, the word "*have*" is omitted in: "...*A validator may ~~have~~ children for....*"

-- On line 14 of page 37 in the Specification, the word "*may*" is misspelled as "*me*": "...*the query me be listed....*"

Appropriate correction is required.

Claim Rejections - 35 USC § 101

35 U.S.C. 101 reads as follows:

Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

2. Claims 31 – 34 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter.

-- Claims 31 – 34 cite a computer-readable medium as the claimed subject matter. The disclosure explicitly states "...Communication media typically embodies computer-readable instructions, data structures, program modules or other data in a modulated data signal such as a carrier wave or other transport mechanism..." and "Combinations

of any of the above should also be included within the scope of computer readable media." on pages 10 and 11. Electromagnetic carrier wave is not statutory because it is a natural phenomenon. In addition, claims 31 and 32 recite various fields of data structure, which is non-functional descriptive material, per se, and directed to an abstract idea. As a result, the subject matter directed to in the claims is non-statutory according to 35 U.S.C 101.

Claim Rejections - 35 USC § 102

The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

3. Claims 1, 3 – 5, 8 – 10, 31, and 32 are rejected under 35 U.S.C. 102(e) as being anticipated by **Beck et al**. (6,968,540).

-- Claim 1.

Beck discloses *a method for testing software, comprising*:

• *a scanning engine for creating a dictionary of types found within executable binaries of a build; and*

(Fig. 3. Page 8: lines 6 – 19; "...executable to executable or binary to binary software translation....". Page 11: lines 1 – 10; "...the virtual machine compiler ...recognizes the class ...method ...class inheritance...".)

• *an execution monitor for invoking a method of a type from the dictionary and storing a result returned by invocation of the method in the dictionary.*

(Fig. 6, page 11: line 58 – page 12: line 36; "...virtual machine language compile in combination with the virtual machine late binding functionality causes the class ...method to be invoked ...store a status return value...".)

-- Claims 3 and 4.

Beck discloses *the method of claim 1 further comprising*:

- *a configurator for adding an article of configuration to a configuration for execution*;
- *a configuration loader for loading a configuration for execution.*

(Fig. 1C, page 6: lines 38 – 67; "...the configuration file can be modified or edited via configuration file editor software ...the configuration file can specify particular events ...directives that specify class attributes and values...".)

-- Claims 5 and 8.

Beck discloses *the method of claim 1 further comprising*:

- *a graphical user interface for viewing a representation of the dictionary*;
- *an editor for editing information of a type in the dictionary.*

(Figs. 1A and 1C, page 7: lines 1 – 18; "...user interface functionality ...enables a user to read, inspect, and modify ...program structures constituting or associated with one or more classes.")

-- Claim 9.

Beck discloses *the method of claim 1 wherein*

- *the scanning engine comprises a dependency manager for tracking type dependencies.*

(Fig. 9, page 14: lines 28 – 35; "The class interceptor ...enables the execution activity of class objects to be monitored by interception and monitoring the path execution associated with their partially or fully inherited methods.")

-- Claim 10.

Beck discloses *the method of claim 5 wherein*

- *the graphical user interface comprises a query builder for creating a query about a type in the dictionary.*

(Page 7: lines 1 – 18; "...read, inspect, and modify ...program structures constituting or associated with one or more classes." For every class and its associated attributes, methods, and inheritance to be displayed on the graphical user interface, the GUI must inherently employ a mechanism (i.e. query) to identify and interpret the data structure encoding that class and subsequently transform it into a viewable format for the user)

-- Claim 31. is a computer product claim containing a data structure for performing the method corresponding to the method of claim 1; Therefore, claim 31 is rejected for the same reason set forth in connection to the rejection of claim 1 above.

-- Claim 32.

Beck discloses *the data structure of claim 31 further comprising*

- *a subset of the set of elements having information for a validator of a type.*

(Fig. 1A, page 5: lines 22 – 60; instrumented classes X, Y, and Z.)

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

4. Claim 7 is rejected under 35 U.S.C. 103(a) as being obvious over **Beck et al.** (6,968,540).

-- Claim 7.

Beck discloses *the method of claim 1*;

Beck, however, does not explicitly disclose *the method further comprising*

- *a database manager for managing configurations and execution runs.*

Beck specifically discloses storing build and instrumented build data in volatile and nonvolatile memories of the system (Fig. 2, page 7: lines 40 – 49). Official notice is taken that database systems had been widely known and used to logically organize related data to ensure its integrity and quality at the time of the invention was made. Thus, it would have been obvious to one having ordinary skill in the art at the time the invention was made to use a database system for storing the configuration and execution result data in order to provide better data organization and manageability for the program application.

5. Claims 2, 6, 11 – 21, 23 – 30, and 33 – 35 are rejected under 35 U.S.C. 103(a) as being unpatentable over **Beck et al**. (6,968,540) in view of **Witchel et al**. (6,748,584).

-- Claim 2.

Beck discloses *the method of claim 1*;

Beck, however, does not disclose the method further comprising

• *a results manager for comparing results returned by two different invocations of a method.*

Witchel discloses a method for comparing two different versions of a text file, comprising:

• *a results manager for comparing results returned by two different invocations of a method.*

(Page 1: lines 7 – 14; "standard textual comparison tools".)

Instrumentation of software system is a technique used to facilitate testing, comprehension and monitoring of operation. The method for software instrumentation of **Beck** can, therefore, be applied in software testing, specifically, regression testing and test coverage analysis. In regression testing of a software program, the results produced by instrumentation of the current version and a previous one of the software can be concretely differentiated to determine the newly made modifications in order to form an updated test plan for the program. Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to enable the software instrumentation method of **Beck** to compare results produced by two different versions

of instrumentation of a method to determine the changes in the method and formulate a testing strategy for the software.

-- Claim 6.

Beck discloses *the system of claim 1*,

Beck, however, does not disclose the method further comprising

• *a delta generator for computing a difference between executable binaries of two software builds.*

Witchel discloses a method comprising:

• *a delta generator for computing a difference between executable binaries of two software builds.*

(Fig. 1, page 5: lines 6 – 42; "Each program binary is "disassembled" ...A test comparison utility ...is then used to produce a list of differences between two disassemblies." Fig. 2, page 6: lines 4 – 14; "...graph matching techniques ...Each source or binary version ...is analyzed and a respective control flow graph representation is produced for each ...The graphs or their representations are compared by a graph matching comparator, and a list of differences is produced.")

Instrumentation of software system is a technique used to facilitate testing, comprehension and monitoring of operation. The method for software instrumentation of **Beck** can, therefore, be applied in software testing, specifically, regression testing and test coverage analysis. In regression testing of a software program, the results produced by instrumentation of the current version and a previous one of the software can be concretely differentiated to determine the newly made modifications in order to

form an updated test plan for the program. Instrumentation reports can further help the user pinpoint code sections of interest in the program being instrumented. Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to enable the software instrumentation method of **Beck** the capability of discovering differences between executable binaries as disclosed in **Witchel** to precisely locate the modified and/or added instructions to develop new test cases to cover these changes.

-- Claim 11: is a method claim for performing the method corresponding to the method of claims 1 and 2; Therefore, claim 31 is rejected for the same reasons set forth in connection to the rejections of claims 1 and 2 above.

-- Claims 12, 13, 14, and 16: are method claims for performing the method corresponding to the method of claims 3, 4, 9, and 8, respectively; Therefore, claims 12, 13, 14, and 16 are rejected for the same reasons set forth in connection to the rejections of claims 3, 4, 9, and 8 above, respectively.

-- Claim 15.

Beck and **Witchel** disclose *the method of claim 11*,

Beck further discloses that method further comprising

- *the step of storing a member of a type in the dictionary.*

(Fig. 1A, page 5: lines 14 – 60; instrumented classes X, Y, and Z are children of program classes A, B, and C)

-- Claim 17.

Beck and **Witchel** disclose *the method of claim 11*,

Beck further discloses that the method further comprising

• *the step of creating an instance of a type from the dictionary for invocation.*

(Fig. 1A, page 5: lines 14 – 60; Creation of instrumented classes X, Y, and Z for intercepting and monitoring execution paths associated with the classes being instrumented.)

-- Claim 18.

Beck and **Witchel** disclose *the method of claim 11*,

Witchel further discloses that the method further comprising

• *the step of generating a listing of code that produced different results from the results of the other build.*

(Figs. 1 and 2, "difference list" identifies changes of code from an older version to a new one)

-- Claim 19.

Beck and **Witchel** disclose *the method of claim 11*,

neither one of them explicitly discloses that

• *the step of reporting the number of blocks of code that executed during invocation of the method.*

Witchel, however, discloses that test coverage monitors program instructions, including program procedures and blocks, that are and are not exercised during a the program

execution (Page 6: lines 27 – 29; "One key application of the present invention is in test coverage analysis used to determine what portions of a software program have been exercised in testing."). Thus, it would have been obvious to one of ordinary skill in the art at the time the invention was made to obtain the number of blocks of executed code during a method invocation when running test coverage as a measurement of program coverage (in addition to the number of instructions executed) in terms of program blocks coverage.

-- Claims 20, 21, and 23.

Beck and **Witchel** disclose *the method of claim 11*,

neither one of them explicitly discloses that

- *the step of reporting the number of branches from blocks of code that executed during invocation of the method*;
- *the step of reporting the number of functions in the code that executed during invocation of the method*;
- *the step of reporting the number of classes in the code that executed during invocation of the method.*

Witchel, however, discloses that the program comparison algorithm uses graph-matching technique (Fig. 2, page 6: lines 4 – 13; "control flow graph representation".) The flow graph contains node and instructional braches showing the relationship between program constructs and executions. Thus, program constructs and structure related to the executed nodes and branches of the program are calculated and monitored by the technique in order to effectively provide test coverage. Therefore, it

would have been obvious to one of ordinary skill in the art at the time the invention was made to obtain, from the flow graph, the numbers of branches from blocks of code, functions, procedures, and classes associated with the methods being executed during a method invocation to provide coverage information on program constructs.

-- Claim 22.

Beck and **Witchel** disclose *the method of claim 11*,

neither one of them explicitly discloses that

• *the step of reporting the number of source files for the code that executed during invocation of the method.*

Witchel, however. discloses that the binary file is an computer-executable file derived from one or more files of high-level source code (Fig. 1, page 3 – 5.). A compiler is inherently capable of identify the related intermediate files (i.e., object files) derived from the source files and proper link them together to form an executable binary file. As the comparator calculates the disassembled binaries to identify the changes of code between different source code versions, the information relating to the source files from which the binary and disassembled files must also be made available to the comparator and consequently, difference list. Thus, it would have been obvious to one of ordinary skill in the art at the time the invention was made to obtain, from the information gathered by the compile, the number of source files for the executed code during a method invocation to provide information on the specific file locations for the executed and unexecuted code.

-- Claim 24.

Beck and **Witchel** disclose *the method of claim 15*,

Beck further discloses that

• *the step of storing members of a type in the dictionary comprises storing a method of the type.*

(Figs. 6 and 7, "Fully overridden method invocation" and "Partially overridden method invocation".)

-- Claim 25.

Beck and **Witchel** disclose *the method of claim 15*,

Beck further discloses that

• *the step of storing members of a type in the dictionary comprises storing a parameter of the type.*

(Fig. 1A, page 6: lines 52 – 67; "...the configuration file can contain directives that specify class attributes and values...".)

-- Claims 26 and 27.

Beck and **Witchel** disclose *the method of claim 15*,

Beck further discloses that

• *the step of storing members of a type in the dictionary comprises storing a validator of the type;*

• *the step of storing members of a type in the dictionary comprises storing a constructor of the type.*

(Fig. 1A, page 5: lines 22 – 60; instrumented classes X, Y, and Z are children of program classes A, B, and C. Instrumented classes provide instrumentation for the program under test.)

-- Claim 28.

Beck and **Witchel** disclose *the method of claim 11*,

Witchel further discloses that

• *the step of generating a report listing any difference between types of the dictionary and types of a dictionary from another build.*

(Figs. 1 and 2, "difference list" identifies changes of code from an older version to a new one, including classes and their associated methods and variables.)

-- Claim 29.

Beck and **Witchel** disclose *the method of claim 11*,

Witchel further discloses that

• *the step of generating a report listing any difference comprises listing any type that is new compared to the dictionary of the other build.*

(Fig. 1, "difference list" and marking of inserted code.)

-- Claim 30.

Beck and **Witchel** disclose *the method of claim 11*,

Witchel further discloses that

• *the step of generating a report listing any difference comprises listing any type that is deleted compared to the dictionary of the other build.*

(Figs. 1 and 2, "difference list" and marking of changed code.)

-- Claim 33.

Beck discloses *a computer-readable medium having computer-executable instructions for performing the steps of*

- *scanning a binary executable from a build*;

(Fig. 3. Page 8: lines 6 – 19; "...executable to executable or binary to binary software translation....".)

- *creating a dictionary of types found in the build from scanning the binary executable*;

(Fig. 3, page 11: lines 1 – 10; "...the virtual machine compiler ...recognizes the class ...method ...class inheritance...".)

- *invoking a method of a type in the dictionary.*

(Fig. 6, page 11: line 58 – page 12: line 36; "...virtual machine language compile in combination with the virtual machine late binding functionality causes the class ...method to be invoked ...store a status return value...".)

Beck, however, does not disclose the further steps of

- *comparing a result of the invocation of the method with a result of an invocation of the method from another build.*

- *executing selective tests for any type that has a result different from the result of an invocation of the method from the other build.*

Witchel discloses the steps of:

• *comparing a result of the invocation of the method with a result of an invocation of the method from another build;*

(Page 1: lines 7 – 14; "standard textual comparison tools".)

• *executing selective tests for any type that has a result different from the result of an invocation of the method from the other build.*

(Fig. 4, page 6: line 51 – page 7: line 10; code markings for changed and unchanged instructions after determination of code differences and test execution.)

Instrumentation of software system is a technique used to facilitate testing, comprehension and monitoring of operation. The method for software instrumentation of **Beck** can, therefore, be applied in software testing, specifically, regression testing and test coverage analysis. In regression testing of a software program, the results produced by instrumentation of the current version and a previous one of the software can be concretely differentiated to determine the newly made modifications in order to form an updated test plan for the program. Subsequently, test cases developed to selectively test the software changes can be run.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to enable the software instrumentation method of **Beck** to compare results produced by two different versions of instrumentation of a method to determine the changes in the method and formulate a testing strategy for the software.

Furthermore, it would have been obvious to enable the software instrumentation method of **Beck** to run selective tests on the changes identified in the instrumentation results to increase test coverage and optimize testing time.

-- Claim 34: is a computer product claim having computer executable components for performing the method corresponding to the method of claims 1 and 2; Therefore, claim 34 is rejected for the same reasons set forth in connection to the rejections of claims 1 and 2 above.

-- Claim 35:

Examiner's Note: it appears that the Applicant is attempting to invoke 35 U.S.C. 112, 6th paragraph, with the use of means-plus-function language in claim 35. However, the specification does not provide any specific physical structure(s) for the features that could be read into the claim to limit the scope of the means to perform the claimed functions. Therefore, the Examiner does not consider the specification to be adequate to invoke a 35 U.S.C. 112, 6th paragraph interpretation and furthermore, for the purpose of further claim analysis under 35 U.S.C. 102 and 103, The Examiner treats claim 35 as a computer program containing machine-readable instructions stored on a physical medium for performing method or the steps recited in the claim.

Claim 35 is a computer product claim having computer executable components for performing the method corresponding to the method of claims 1 and 2; Therefore, claim 35 is rejected for the same reasons set forth in connection to the rejections of claims 1 and 2 above.

Conclusion

The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

-- **Berry et al.** (US 6,742,178), System and Method for Instrumenting Application Class Files with Correlation Information to the Instrumentation. IBM Corporation.

-- **Burridge et al.** (US 6,918,106), Method and Apparatus for Collocating Dynamically Loaded Program Files. Sun Microsystems, Inc.

-- **Koved et al.** (US 6,925,638), Mutability Analysis in Java. IBM Corporation.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Thai Van Pham whose telephone number is (571) 270-1064. The examiner can normally be reached on Monday - Thursday, 9am - 5pm EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Wei Y. Zhen can be reached on (571) 272-3708. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

TVP
10/23/2006

Wei Y. Zhen
Supervisory Patent Examiner